DETAILED ACTION

This Office Action is in response to the communication filed on 07/21/2008.

Claims 26-37 have been amended.

Claims 1-37 have been examined and are pending.

Response to Arguments

The objections of the specification, see page 11, filed 07/21/2008 have been withdrawn in view of amendment.

Applicant's arguments, see pages 11-12, filed 07/21/2008 have been fully considered but they are not persuasive.

The Applicant argues the following:

(A) "Zimmer 4687 and the present application are assigned to common assignee (Intel Corporation). As such, Zimmer 4687 is an invalid reference under 35 USC 102 ."

(B) "Zimmer 4687, Zimmer 1968 and the present application are all assigned to a common assignee (Intel Corporation) ."

The Examiner respectfully disagrees as the following reasons:

<u>Per (A):</u>

According to section **MPEP 716.10 (e)** (page 349 in MPEP):

> In the situation described in Example 2, an affidavit under 37 CFR 1.132 may be submitted to show that the relevant portions of the reference originated with or were obtained from applicant. Thus the affidavit attempts to convert the fact situation from that described in Example 2 to the situation described in Example 1.

It requires that the Applicant submits an affidavit under 37 CFR 1.132. Therefore, the 35 U.S.C. 102(e) rejection of claims 20-25 is maintained.

<u>Per (B):</u>

According to section **MPEP 706.02 (l) (1) [R-6]** (page 62 in MPEP):

> (c)
> (1) Subject matter developed by another person, which qualifies as prior art only under one or more of subsections (e), (f), and (g) of section 102 of this title, shall not preclude patentability under this section where the subject matter and the claimed invention were, at the time the claimed invention was made, owned by the same person or subject to an obligation of assignment to the same person.

> The burden of establishing that subject matter is disqualified as prior art is placed on applicant once the examiner has established a *prima facie* case of obviousness based on the subject matter. For example, the fact that the reference and the application have the same assignee is <u>not</u>, by itself, sufficient evidence to disqualify the prior art under 35 U.S.C. 103(c). There must be a statement that the common ownership was "at the time the invention was made."

It requires that the Applicant must include a statement that the common ownership was **"at the time the invention was made".** Therefore, the rejection 35 U.S.C. 103(a) of claims 1-19 and 26-37 is maintained.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 20-25 are rejected under 35 U.S.C. 102(e) as being anticipated by Zimmer (20050114687 A1) hereinafter **Zimmer 4687.**

<u>As per claim 20:</u>

Zimmer 4687 discloses a system for monitoring software integrity, comprising:

a trusted computing device **[par. [0028], line 4; an example security module includes a trusted platform module (TPM)];**

a protected partition machine running on the trusted computing device **[figure 2, trusted partition 204 and protected memory 226];**

a guest virtual machine running on the trusted computing device, the guest virtual machine including guest software **[figure 2, Apps 212 and OS 216];**

a secure memory area on the trusted computing device **[figure 2, protected memory 226];** and

an integrity monitor executing within the protected partition, the integrity monitor configured to monitor the guest software in the guest virtual machine running on the trusted computing device and to take action if the guest software is deemed comprised, , the integrity monitor capable of generating a baseline hash value for the guest software initially, and a current hash value for the guest software during runtime, the integrity monitor further capable of storing the baseline hash value in the secure memory area, the integrity monitor further capable of comparing the baseline hash value and the current hash value to determine if the guest software has been compromised **[par. [0031], lines 18-19, par. [0032], lines 1-8; figure 2; par. [0040], lines 1-6; "The SVMM 110 is configured to communicate with the untrusted partition 202 and the trusted partition 204 such as, for example, the protected firmware resources 106, the operating system 216, the SMM runtime firmware 218, and the secure operating system 224. The SVMM 110 is a trusted and secure kernel, thus runs at ring(0-1) 206"; par. [0026], lines 5-19; "The secure kernel 110 may be a secure virtual machine monitor (SVMM) that is securely launched during or after a boot phase of an operating system. A person of ordinary skill in the art will readily appreciate that the SVMM 110 is a known secure application that is typically used to establish and enforce security/protection policies. For example, during a boot phase, the SVMM 110 may establish protection policies for its resources and the resources of other protected applications such as a protected operating system. The SVMM 110 may also establish and enforce a firmware resource protection policies for the protected firmware resources 106 specified in the resource protection list 108. Additionally, the SVMM 110 may be configured to monitor processor system and software/firmware behavior to ensure safe and secure operation"].**

<u>As per claim 21:</u>

Zimmer 4687 also discloses the system according to claim 20 wherein the secure memory area includes a trusted platform module ("TPM") **[par. [0028], line 4; an example security module include a trusted platform module (TPM)].**

<u>As per claim 22:</u>

Zimmer 4687 also disclose the system according to claim 20 wherein the trusted computing device may calculate a hash value for the integrity monitor and store the hash valuefor the integrity monitor in the secure memory area **[par. [0049], lines 6-8; hashing each protection descriptor and storing the hash code in a secure register such as, for example a TPM PCR].**

<u>As per claim 23:</u>

Zimmer 4687 also disclose The system according to claim 22 wherein the hash value for the integrity monitor may be used to verify the integrity monitor prior to enabling the integrity monitor to access the baseline hash value stored in the secure memory area **[figure 2; par. [0029]; par. [0032]; "each hash code may be used to validate it respective protection descriptor stored in the resource protection list 108"].**

<u>As per claim 24:</u>

Zimmer 4687 also disclose the system according to claim 21 wherein the trusted computing device executes in Secure Execution Machine ("SMX") mode and the secure memory area includes one of the TPM and a designated non-writable memory area **[A resource protected list can be stored in a TPM, PCR (par. [0049], lines 4-9). A protected firmware resource 106 is stored in the protected memory 226 (i.e., a non-writable memory); figure 1, figure 2; par. [0041], lines 1-12].**

As per claim 25:

Zimmer 4687 also disclose the system according to claim 24 wherein a secure launch module may calculate a hash value for the integrity monitor **[par. [0029], "The secure encryption functions may enable a hashing function to encrypt each protection descriptor...Computer Systems Laboratory"]** and store the hash value for the integrity monitor in the secure memory area **[par. [0029]; after hashing each protection descriptor, the hash codes may be store in PCRs].**

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-19 and 26-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Zimmer (US 20050021968 A1) hereinafter **Zimmer 1968** in view of Zimmer (US 20050114687 A1) hereinafter **Zimmer 4687**.

As per claim 1:

Zimmer 1968 discloses a method of monitoring software executing on a trusted computing device comprising:

Executing an integrity monitor in a protected on the trusted computing device, the integrity monitor configured to monitor guest software in a guest virtual machine on the trusted computing device and to take action if the guest software is deemed comprised **[par. [0026], lines 5-19; "The secure kernel 110 may be a secure virtual machine monitor (SVMM) that is securely launched during or after a boot phase of an operating system. A person of ordinary skill in the art will readily appreciate that the SVMM 110 is a known secure application that is typically used to establish and enforce security/protection policies. For example, during a boot phase, the SVMM 110 may establish protection policies for its resources and the resources of other protected applications such as a protected operating system. The SVMM 110 may also establish and enforce a firmware resource protection policies for the protected firmware resources 106 specified in the resource protection list 108. Additionally, the SVMM 110 may be configured to monitor processor system and software/firmware behavior to ensure safe and secure operation")].**

storing the baseline values in a secure memory area **[par. [0019], lines 1-6; A baseline level of security is equivalent to a baseline values].**

processing the guest software during runtime according to a predefined methodology to determine current runtime information **[par. [0050]; A firmware1 is equivalent to software in a context of the prior. The firmware update drive issues a SENDER command upon execution (i.e. runtime). A processor hash-extends a binary image of update driver using hash algorithm SHA-1 (i.e. predefined methodology)].**

comparing the current runtime information to the baseline values stored in the secure memory area to determine whether the guest software has been compromised **[par. [0035], lines**

7-9; "comparing integrity metrics corresponding to a current platform environment with the given platform environment"].

Zimmer 1968 does not disclose generating in the protected partition on the trusted computing device baseline values.

However, Zimmer 4687 discloses generating in the protected partition on the trusted computing device baseline values pertaining to guest the software in the guest virtual machine **[A protected operating partition is equivalent to a trusted partition (par. [0031], lines 11-19). A code in trusted partition can be setup to run at different operating modes or privilege levels of processor (par. [0032], lines 1-3). As a result, some metadata information are generated from this protected partition relate to guest software. Virtual machine monitor (VMM) presents to other software (i.e. "guest" software) the abstraction of one or more virtual machines (VMs). In figure 2, Secure virtual machine monitor (SVMM) 110 presents other software (e.g., OS 216 or 212 APPS) as guest software].**

Therefore, it would have been obvious to the person of ordinary skill in the art at the time the invention was made to modify the method of Zimmer 1968 by including the step of Zimmer 4687 because it would perform system resource verification tests to ensure proper resource configuration/operation **[par. [0005], lines 13-14, Zimmer 4687].**

[1] Definition of firmware on page 438 of IEEE 100, THE AUTHORITATIVE DICTIONARY OF IEEE STANDARDS TERMS, SEVEN EDITION, IEEE Published by Standards Information Network IEEE Press, ISBN 0-7381-2601-2

<u>As per claim 2:</u>

Zimmer 4687 further discloses performing a hash function on the guest software to obtain a hash value **[par. [0029], lines 1-4; "The secure encryption functions may enable a hash function to encrypt each protection descriptor as a hash code; A firmware is a software. The pre-boot firmware 102 in the pre-boot environment 100 based on the content descriptors may include any firmware resource par. [0022], lines 4-6].**

<u>As per claim 3:</u>

Zimmer 4687 further discloses performing the hash function on the guest software includes performing a hash function on one of each component of the guest software and a collection of components of the guest software **[par. [0029], lines 1-3, par. [0050], lines 19-20. Each protection descriptor is equivalent to each component of the guest software; descriptors stored in the resource protection list (RPL) 108 are equivalent to a collection of component of the guest software].**

<u>As per claim 4:</u>

Zimmer 1968 also discloses the method according to claim 2 wherein performing the hash function on the guest software to obtain the hash value further comprises at least one of performing the hash function on the guest software prior to execution to obtain an initial static baseline value and performing the hash function on the guest software immediately upon execution to obtain an initial runtime baseline value **[A cryptographic hash is equivalent to**

hash function prior to execution (par. [0032], lines 1-3). Performing the hash function on the guest software immediately upon execution to obtain an initial runtime baseline is equivalent to performing a hash operation on the firmware update driver (par. [0025], lines 5-6)].

<u>As per claim 5:</u>

Zimmer 1968 also discloses the method according to claim 4 wherein processing the guest software during runtime according to a predefined methodology further comprises performing the hash function periodically on the guest software during runtime to obtain a current hash value **[Performing the hash function periodically on the guest software is equivalent to performing a hash operation on the firmware update driver (par. [0025], lines 5-6)].**

<u>As per claim 6:</u>

Zimmer 1968 also discloses the method according to claim 5 wherein comparing the current runtime information to the baseline values further comprises comparing the current hash value to the baseline hash value **[par. [0035], lines 7-9; comparing integrity metrics corresponding to a current platform environment with the given platform environment. The current hash value to the baseline hash value is equivalent to current platform environment with the given platform environment].**

<u>As per claim 7:</u>

Zimmer 1968 also discloses the method according to claim 1 wherein generating the baseline values comprises retrieving the baseline values from a storage location on the trusted computing device **[Retrieving the baseline values is equivalent to retrieve key and certificate data (par. [0015], lines 5-9). A trusted computing device is equivalent to a TPM 126 (par. [0018])].**

<u>As per claim 8:</u>

Zimmer 1968 also discloses the method according to claim 1 wherein storing the baseline values in the secure memory area further comprises storing the hash value in a trusted platform module ("TPM") **[par. [0019], lines 1-6; "A TPM provides a baseline level of security for storing and accessing trusted-related data and authentication measures"].**

<u>As per claim 9:</u>

Zimmer 1968 does not disclose performing a secure launch of the trusted computing platform prior to generating the baseline values.

However, Zimmer 4687 discloses performing a secure launch of the trusted computing platform prior to generating the baseline values **[par. [0050], lines 13-15; "In general, the example boot process 500 may be used to boot/launch a secure kernel"; par. [0052], lines 6-8].**

Therefore, it would have been obvious to the person of ordinary skill in the art at the time the invention was made to modify the method of Zimmer 1968 by including the step of Zimmer 4687 because it would provide for future authentication or validation of trustworthiness **[Zimmer 4687, par. [0056], lines 11-14].**

<u>As per claim 10:</u>

Zimmer 1968 does not explicitly disclose storing the hash value in one of a TPM and a designated non-writable memory area.

However, Zimmer 4687 discloses storing the baseline values in the secure memory area further comprises storing the hash value in one of a TPM and a designated non-writable memory area [Storing the hash code in a secure register such as a TPM PCR. **(par. [0049], lines 7-8).** A protected memory is equivalent to non-writable memory **(par. [0041], lines 10-17)].**

Thus, it would have been obvious to the person of ordinary skill in the art at the time of the invention was made to modify the method of Zimmer 1968 by including the step of Zimmer 4687 because it would be protected from malignant modifications **[Zimmer 4687, par. [0041], lines 10-12].**

<u>As per claim 11:</u>

Zimmer 4687 further discloses executing at least a portion of the guest software in a designated non-writable memory area **[A protected memory is equivalent to non-writable memory (par. [0041], lines 10-17; "a code runs in the protected memory 226 may be protected from being viewed or modified by unauthorized applications")**

<u>As per claim 12:</u>

Zimmer 1968 also discloses the method according to claim 1 wherein the predefined methodology includes at least one of a checksum, MD5 and SHA1 **[par. [0050], lines 5-7; a same hash algorithm is employed such as the SHA-1 hash algorithm].**

<u>As per claim 13:</u>

Zimmer 1968 does not disclose the protected partition wherein includes a root virtual machine.

However, Zimmer 4687 discloses the protected partition wherein includes a root virtual machine **[A trusted partition (i.e., a protected operating partition) is equivalent to the protected partition. The trusted partition includes a secure OS and a secure virtual machine monitor (SVMM) (figure2; par. [0026], lines 5-7; par. [0031], lines 18-19].**

Thus, it would have been obvious to the person of ordinary skill in the art at the time the invention was made to modify the method of Zimmer 1968 by including the step of Zimmer 4687 because a trusted protected (i.e., a protected operating partition) includes a root virtual machine **[Zimmer 4687, figure 2].**

<u>As per claim 14:</u>

Zimmer 1968 discloses a method of monitoring the integrity of a trusted computing device, comprising:

storing the baseline value in a secure memory area **[par. [0019], lines 1-6; A baseline level of security is equivalent to a baseline values]**;

the integrity monitor periodically processing the guest software while executing to generate a current hash value **[par. [0025], lines 5-6; "this integrity metric is generated by performing a hash function operation on the firmware update driver"]**; and the integrity monitor comparing the baseline hash value in the secure memory area to the current hash value to determine whether the guest software has been compromised **[par. [0035], lines 7-9; "comparing integrity metrics corresponding 1to a current platform environment with the given platform environment"];**

Zimmer 1968 does not disclose "launching a protected partition and a guest virtual machine on the trusted computing device, executing an integrity monitor in the protected partition and guest software in the guest virtual machine, and processing the guest software in the guest virtual machine".

However, Zimmer 4687 discloses:

launching a protected partition and a guest virtual machine on the trusted computing device **[A protected partition is equivalent to a trusted partition (i.e. a protected operating partition). The trusted partition may be configured (i.e., set up) to run at different operating modes or privilege levels of processors. It means for the protected partition and guest virtual machine (par. [0031], lines 18-19; par. [0032], lines 1-8; figure 2)];**

executing an integrity monitor in the protected partition and guest software in the guest virtual machine, the integrity monitor configured to monitor guest software in the guest virtual machine on the trusted computing device and to take action if the guest software is deemed comprised **[A secure virtual machine monitor (i.e., secure kernel) is equivalent to the integrity monitor. It runs at different operating modes or privilege levels of processors. It means to execute in the protected partition and guest software in the guest virtual machine (par. [0031], lines 18-19, par. [0032], lines 1-8; figure 2; par. [0040], lines 1-6; par. [0026], lines 5-19**; **"The secure kernel 110 may be a secure virtual machine monitor (SVMM) that**

is securely launched during or after a boot phase of an operating system. A person of ordinary skill in the art will readily appreciate that the SVMM 110 is a known secure application that is typically used to establish and enforce security/protection policies. For example, during a boot phase, the SVMM 110 may establish protection policies for its resources and the resources of other protected applications such as a protected operating system. The SVMM 110 may also establish and enforce a firmware resource protection policies for the protected firmware resources 106 specified in the resource protection list 108. Additionally, the SVMM 110 may be configured to monitor processor system and software/firmware behavior to ensure safe and secure operation"]

the integrity monitor processing the guest software in the guest virtual machine to generate a baseline hash value **[In figure 2, Secure virtual machine monitor (SVMM) 110 presents other software (e.g., OS 216 or 212 APPS) as guest software. A guest software runs on a guest virtual machine; par. [0029], lines 1-3 and lines 9- 12, "the secure encryption function may enable a hashing function to encrypt each protection descriptor as hash code (i.e. hash value)"].**

Therefore, it would have been obvious to the person of ordinary skill in the art at the time the invention was made to modify the method of Zimmer 1968 by including the step of Zimmer 4687 because it would perform system resource verification tests to ensure proper resource configuration/operation **[par. [0005], lines 13-14, Zimmer 4687].**

<u>As per claim 15:</u>

Zimmer 1968 does not disclose storing the baseline value in a secure memory area including storing the baseline value in at least one of a trusted platform module ("TPM") and a designated non-writable memory area.

However, Zimmer 4687 discloses the method according to claim 14 wherein storing the baseline value in a secure memory area includes storing the baseline value in at least one of a trusted platform module ("TPM") and a designated non-writable memory area **[Storing the hash code in a secure register such as a TPM PCR. (par.[0049], lines 7-8). A protected memory is equivalent to non-writable memory (par.[0041], lines 10-17)].**

Thus, it would have been obvious to the person of ordinary skill in the art at the time of the invention was made to modify the method of Zimmer 1968 by including the step of Zimmer 4687 because it would because it would be protected from malignant modifications **[Zimmer 4687, par. [0041], lines 10-12].**

<u>As per claim 16:</u>

Zimmer 1968 also discloses processing and storing a value corresponding to the integrity monitor **[par. [0035], lines 5-6, storing integrity metrics corresponding to a given platform environment].**

<u>As per claim 17:</u>

7immer 1968 also discloses the method according to claim 16 further comprising verifying the integrity monitor prior to comparing the baseline hash value to the current hash value **[An integrity metric is generated by performing a hash operation on the firmware (par. [0025], lines 3-6). The current hash value to the baseline hash value is equivalent to current**

platform environment with the given platform environment; par. [0035], lines 7-9, comparing integrity metrics corresponding to a current platform environment].

<u>**As per claim 18:**</u>

Zimmer 1968 does not disclose processing the guest software in the guest virtual machine to generate the baseline hash value includes retrieving the baseline hash value from a storage location.

However, Zimmer 4687 also discloses processing the guest software in the guest virtual machine to generate the baseline hash value includes retrieving the baseline hash value from a storage location **[par. [0029], lines 9-12; "the hash codes may be stored in PCRs in the pre-boot environment 100 by the pre-boot firmware 102 and retrieved from the PCRs in the post-boot environment 101 by the SVMM110"].**

Therefore, it would have been obvious to the person of ordinary skill in the art at the time of the invention was made to modify the method of Zimmer 1968 by including the step of Zimmer 4687 because it would perform system resource verification tests to ensure proper resource configuration/operation **[par. [0005], lines 13-14, Zimmer 4687].**

<u>**As per claim 19:**</u>

Zimmer 1968 does not disclose launching a protected partition includes launching a root virtual machine.

Zimmer 4687 discloses launching a protected partition includes launching a root virtual machine **[A trusted partition (i.e., a protected operating partition) is equivalent to the**

protected partition. The trusted partition includes a secure OS and a secure virtual machine monitor (SVMM) (figure 2; par. [0026], lines 5-7; par.[0031], lines 18-19].

Thus, it would have been obvious to the person of ordinary skill in the art at the time the invention was made to modify the method of Zimmer 1968 by including the step of Zimmer 4687 because a trusted protected (i.e., a protected operating partition) includes a root virtual machine **[Zimmer 4687, figure 2].**

Claim 26 is essentially the same as claim 1 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 27 is essentially the same as claim 2 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 28 is essentially the same as claim 3 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 29 is essentially the same as claim 4 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 30 is essentially the same as claim 5 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 31 is essentially the same as claim 6 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 32 is essentially the same as claim 7 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 33 is essentially the same as claim 8 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 34 is essentially the same as claim 9 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 35 is essentially the same as claim 10 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 36 is essentially the same as claim 11 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Claim 37 is essentially the same as claim 13 except that it sets forth the claimed invention as an article comprising a medium accessible by a trusted computing device rather a method and rejected under the same reasons as applied above.

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Canh Le whose telephone number is 571-270-1380. The examiner can normally be reached on Monday to Friday 7:30AM to 5:00PM other Friday off.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Zand Kambiz can be reached on 571-272-3811. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Canh Le/

Examiner, Art Unit 2439

October 22, 2008
/Kambiz Zand/
Supervisory Patent Examiner, Art Unit 2434